DPAS: Demand-Pressure Adaptive Slicing for Real-Time 6G Network Resource Management

Jaynil D Panjwani*, Neha Agrawal†
*Dept. of ECE, IIIT Sri City, Chittoor, India
†Dept. of CSE, IIIT Sri City, Chittoor, India
Email: jaynil.panjwani13@gmail.com, neha.a@iiits.in

***Abstract*—Next-generation 6G base stations must jointly serve eMBB, URLLC, and mMTC traffic within sub-millisecond scheduling windows under strict and conflicting latency constraints. Existing methods either neglect URLLC urgency or fail to enforce per-slice bandwidth floors. We propose DPAS (Demand-Pressure Adaptive Slicing), a closed-form, zero-training scheduler that represents each slice as a weighted demand-pressure ratio $P_i^w = w_i D_i / B_i$ and reallocates bandwidth via a lightweight gradient update with a floor-preserving two-stage projection. The algorithm runs in $\mathcal{O}(N)$ per slot, requires no offline training, and—under our linearised model and simulations—typically settles within 6–8 slots. On 3.67 million real-world CDRs across seven spatial zones, DPAS achieves above 99% queueing-delay SLA for all three slice types.**

***Index Terms*—6G, network slicing, demand pressure, Lyapunov stability, SLA, zero-training, resource block allocation, 5G NR.**

I. Introduction

6G base stations must simultaneously carry latency-critical control traffic (URLLC), massive sensor data (mMTC), and high-throughput video (eMBB) over a shared radio channel, with each allocation decision made within 1 ms [1]. The challenge is to maintain per-slice SLA guarantees under unpredictable, asymmetric load—not merely to achieve coexistence.

Proportional fairness performs well on average but is structurally blind to deadline asymmetry: URLLC and mMTC can carry comparable mean demand yet have latency budgets differing by orders of magnitude, so a fair scheduler treats them identically. When eMBB surges, bandwidth shifts toward it and squeezes the remaining slices; equal demand does not imply equal urgency.

None of the existing solutions resolves this cleanly. Static allocation can absorb bursts but cannot adapt. Deep reinforcement learning [2] requires hours of offline training and 5–15 ms inference latency, which exceeds a URLLC deadline. Lyapunov Drift-Plus-Penalty [3] offers formal stability but solves a linear program per slot; at 5G NR resource-block scales this can exceed the 1 ms budget [4]. Model Predictive Control (MPC) [5] can be fast on average but degrades under the heavy-tail bursts where accurate decisions matter most.

DPAS addresses this by representing each slice as a weighted demand pressure $P_i^w = w_i D_i / B_i$, combining proportional-fair normalisation with Lyapunov-inspired pressure balancing while avoiding a per-slot optimisation problem. A two-stage floor projection enforces per-slice minima in closed form; both the update and the projection remain strictly $\mathcal{O}(N)$, keeping DPAS within sub-millisecond scheduling budgets.

A. Contributions

The main contributions of this work are:

- We introduce a weighted demand-pressure metric P_i^w that jointly captures slice congestion and service urgency in a single closed-form scalar, and derive a zero-training online slicing algorithm with $\mathcal{O}(N)$ per-slot complexity requiring no offline traffic model.
- We design a two-stage floor-preserving projection that guarantees feasibility under per-slice minimum and maximum bandwidth constraints at every slot.
- Evaluated on 3.67 million real CDRs across seven spatial zones, DPAS achieves above 99% SLA for eMBB, URLLC, and mMTC without per-zone retuning.

II. Related Work

Network slicing was formalised in 3GPP Release 15 and has since attracted extensive work on architectures and resource-management mechanisms [6]. Kelly's proportional-fairness framework [7] requires no training and converges to a Pareto-optimal operating point under static load, but it encodes no per-slice deadline urgency and therefore struggles under bursty eMBB traffic. Sciancalepore et al. [5] showed that MPC traffic forecasting degrades precisely at the heavy-tail events where accurate decisions matter most, while deep RL schedulers [2] require hours of offline training, incur 5–15 ms inference latency, and can degrade silently under distribution shift. The closest theoretical antecedent to DPAS is Neely's stochastic network optimisation [3], which provides Lyapunov-based stability guarantees without requiring a traffic model. DPAS inherits this pressure-balancing view but replaces the per-slot linear program [4] with a closed-form $\mathcal{O}(N)$ update that directly encodes per-slice deadline urgency in the pressure signal. Lightweight mechanisms such as weighted round-robin and max-weight scheduling are compatible with this system model; we focus on proportional fairness and Lyapunov drift-plus-penalty as the most widely studied slice-level mechanisms with known stability properties, and leave a broader baseline sweep to future work. Table II summarises how these paradigms compare against the deployment criteria defined in Section I.

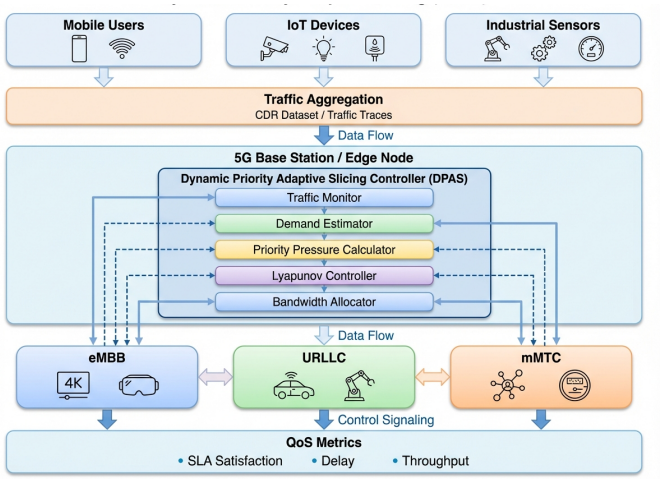

Fig. 1: DPAS control loop inside a 6G gNodeB. Demand Observer → DPAS Controller ($\mathcal{O}(N)$, est. $<$1 ms) → RB Allocator, within one 100 ms NR frame.

TABLE I: Per-slice bandwidth bounds, SLA delay thresholds, and urgency weights.

Slice	$B_i^{\min}$	$B_i^{\max}$	τ_i	w_i^*
eMBB	5 Mbps	80 Mbps	100 ms	2.0
URLLC	10 Mbps	50 Mbps	10 ms	1.5
mMTC	5 Mbps	20 Mbps	500 ms	1.5

III. System Model

At each 100 ms 5G NR frame boundary, the gNodeB executes the three-stage loop shown in Fig. 1. The Demand Observer reads per-slice MAC queue-fill counters to obtain $D_i(t)$ at zero additional overhead, since the same counters serve per-user scheduling. The DPAS Controller computes $\mathbf{B}(t{+}1)$ in closed form; its $\mathcal{O}(N)$ structure is estimated to complete within the 1 ms scheduling deadline [1], pending hardware validation. The Resource Block Allocator then maps bandwidths to physical resource blocks. Only $\mathbf{B}(t)$ is retained across slots; the controller carries no history and maintains no model.

The allocation $\mathbf{B}(t) \in \mathbb{R}^N$ must satisfy a capacity constraint and per-slice bounds drawn from 3GPP specifications [1], [8] (Table I):

$$\textstyle\sum_i B_i(t) = C, \qquad B_i^{\min} \le B_i(t) \le B_i^{\max}, \tag{1}$$

where $C = 100$ Mbps is the shared channel capacity. Per-slot queuing delay follows the M/M/1 model [9], with $\delta_i = D_i/(B_i - D_i)$; the SLA requires $\delta_i \le \tau_i$ for $\boldsymbol{\tau} = [100,\ 10,\ 500]$ ms. These figures are queueing-model-based and do not represent end-to-end deployment measurements.

IV. The DPAS Algorithm

A. Weighted Demand Pressure

Raw demand alone is an unreliable indicator of urgency. A slice carrying 10 Mbps against an allocation of 12 Mbps is critically congested; the same demand against 80 Mbps is comfortably served. The ratio D_i/B_i captures congestion level, and multiplying by urgency weight w_i ensures that time-critical slices receive higher priority at equal load. DPAS therefore defines the *weighted demand pressure*:

$$P_i^w(t) = \frac{w_i\, D_i(t)}{B_i(t)}, \tag{2}$$

where $w_i > 0$ is drawn from Table I. High P_i^w signals a congested, time-critical slice; low P_i^w signals spare capacity. The system reaches a stable operating point when all pressures equalise.

TABLE II: Five allocation paradigms vs. five deployment criteria. *Lyap.-style stab.*: local Lyapunov-style analysis. *No-Train*: from slot 1. *Fast*: $\le$1 ms. *Floor*: per-slice min. every slot. $\ge$99%: all three slices.

Method	**Lyap.-style stab.**	**No-Train**	**Fast**	**Floor**	$\ge$99%
Static	✗	✓	✓	✗	✗
Prop. Fair [7]	✗	✓	✓	✗	✗
Deep RL [2]	✗	✗	✗	✗	✗
Lyap. D+P [3]	✓	✓	✗	✗	—
MPC [5]	✗	✗	✓	✗	✗
DPAS (ours)	✓	✓	✓	✓	✓

B. C1: Gradient Update

We minimise the surrogate objective $\sum_i (P_i^w - \bar{P}^w)^2$, which penalises pressure imbalance across slices. Differentiating (2) with respect to B_i gives $\partial P_i^w/\partial B_i = -P_i^w/B_i < 0$, confirming that more bandwidth always reduces a slice's pressure. A gradient-like step on this surrogate [4] yields the per-slot update:

$$B_i(t{+}1) = B_i(t) + \alpha\,(P_i^w - \bar{P}^w)\,B_i(t), \tag{3}$$

where $\bar{P}^w = \frac{1}{N}\sum_j P_j^w$ is the mean pressure and $\alpha \in (0, 1/\bar{P}^w)$ is the step size (selected in Section IV-F). Slices with $P_i^w > \bar{P}^w$ receive additional bandwidth; those below the mean yield it.

C. C2: Two-Stage Floor Projection

The output of (3) may not sum to C and may violate minimum limits. Simple renormalisation is insufficient: during an eMBB spike, $\sum B_j' > C$, and scaling every slice uniformly can push mMTC below its minimum allocation. DPAS instead reserves all floor allocations first and distributes the remaining surplus proportionally. Let $R_c = C - \sum_j B_j^{\min} = 80$ Mbps denote the distributable surplus. The two stages are:

$$\tilde{B}_i = \mathrm{clip}(B_i',\ B_i^{\min},\ B_i^{\max}) \tag{S1}$$

$$e_i = \max(0,\ \tilde{B}_i - B_i^{\min}) \tag{S2a}$$

$$B_i = B_i^{\min} + e_i \cdot R_c / \textstyle\sum_j e_j \tag{S2b}$$

S1 clips each candidate into $[B_i^{\min}, B_i^{\max}]$. S2a computes the earned excess e_i above the floor. S2b sets the final allocation as the floor plus a surplus share proportional to e_i. Section V proves that (1) holds for any C1 output.

D. Priority Weights

The weights w_i ensure that a tight-deadline slice generates stronger pressure than a loose-deadline one at equal load. The

TABLE III: Grid search over $B_e^{\max}$ at fixed $w_e = 2.0$, $w_u = w_m = 1.5$ (3,000-slot run).

$B_e^{\max}$	Overall SLA	eMBB SLA
80 Mbps	99.7%	99.3%
82 Mbps	99.8%	99.4%
85 Mbps	99.8%	99.5%
87 Mbps	99.8%	99.5%
90 Mbps	99.8%	99.5%

Algorithm 1 DPAS: Demand-Pressure Adaptive Slicing

Require: $\{D_i(t)\}$; C; α; $\mathbf{w}^*$; $\mathbf{B}^{\min}, \mathbf{B}^{\max}$; $\boldsymbol{\tau}$; $\varepsilon > 0$

$\mathbf{B}(0) \leftarrow [C/N, \ldots, C/N]$ *equal split; no training needed*

$R_c \leftarrow C - \sum_j B_j^{\min}$ *surplus = 80 Mbps*

for each slot t **do**

  $P_i^w \leftarrow w_i^* D_i / \max(B_i, \varepsilon)$ $\forall i$

  $\bar{P}^w \leftarrow \frac{1}{N} \sum_j P_j^w$

  $B_i' \leftarrow B_i + \alpha\,(P_i^w - \bar{P}^w)\,B_i$ $\forall i$

  $\tilde{B}_i \leftarrow \mathrm{clip}(B_i', B_i^{\min}, B_i^{\max})$ $\forall i$

  $e_i \leftarrow \max(0,\, \tilde{B}_i - B_i^{\min})$ $\forall i$

  $\mathbf{B}(t{+}1) \leftarrow \mathbf{B}^{\min} + \mathbf{e} \cdot R_c / \sum_j e_j$

  $\mathbf{B}(t{+}1) \leftarrow \mathrm{clip}(\mathbf{B}(t{+}1), \mathbf{B}^{\min}, \mathbf{B}^{\max})$

  $\delta_i \leftarrow D_i / (B_i - D_i)$; $\not\Vdash_i \leftarrow [\delta_i \le \tau_i]$ $\forall i$

end for

mMTC weight is particularly sensitive: reducing w_m to 0.5 drops mMTC SLA below 70%, because mMTC's low mean demand ($\bar{D}_m = 0.59$ Mbps) produces a weak pressure signal during eMBB bursts. Setting $w_m = 1.5$ amplifies this signal sufficiently to hold mMTC above its floor during peak load.

A coarse grid search over $w_e \in [2.0, 5.0]$ and $B_e^{\max} \in [80, 90]$ Mbps on a 3,000-slot subset showed that, once URLLC and mMTC weights are fixed to $w_u = w_m = 1.5$, overall SLA is largely insensitive to w_e; eMBB performance is governed primarily by $B_e^{\max}$. Table III reports results at $w_e = 2.0$ around the final operating point, confirming >99.7% overall SLA; the full evaluation uses $B_e^{\max} = 80$ Mbps since the 99% target is already met at that bound. The weights do not encode latency strictness directly: URLLC already benefits from a tighter delay target and a larger bandwidth floor, while $w_e > w_u$ compensates for eMBB's heavier burst profile in this dataset without degrading the other slices. Final configuration: $\mathbf{w}^* = [2.0, 1.5, 1.5]$, $B_e^{\max} = 80$ Mbps.

E. Complete Algorithm

The C1, C2 and the weight configuration are combined in a single forward pass in Algorithm 1. The uniform initial allocation and the surplus $R_c = 80$ Mbps, two constants, are pre-computed once before the loop. After $S2$, a safety clip absorbs floating-point rounding on the boundaries of the constraints and a guard $\varepsilon = 10^{-9}$ avoids division by zero on idle slots.

F. Complexity and Step-Size

Each slot requires $\mathcal{O}(N)$ scalar operations: C1 takes a fixed number of arithmetic steps per slice, and C2 takes an additional fixed number per slice plus one pass to compute $\sum_j e_j$. Memory is $\mathcal{O}(N)$ words for the current allocation vector and a few precomputed scalars. Table IV places this in context against all baselines.

TABLE IV: Per-slot complexity and latency. DPAS latency estimated from operation count; no hardware implementation has been performed. DRL from [2]; D+P reflects LP cost at 5G NR RB count [3], [4].

Method	**Ops**	**Memory**	**Latency**	**Stable**
Static	$\mathcal{O}(1)$	$\mathcal{O}(N)$	<0.1 ms	✗
Prop. Fair [7]	$\mathcal{O}(N)$	$\mathcal{O}(N)$	<0.1 ms	✗
Deep RL [2]	$\mathcal{O}(N^2)$	$\mathcal{O}(10^6)$	5–15 ms	✗
Lyap. D+P [3]	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$	>1 ms	✓
DPAS (est.)	$\mathcal{O}(N)$	$\mathcal{O}(N)$	<1 ms*	✓

*Estimated from operation count; hardware validation is future work.

The step size α must satisfy $\alpha < 1/\bar{P}^w$ for the Lyapunov-style contraction to hold. Across the seven zones, the stability ceiling spans 0.56–1.43 with $\bar{P}^w$ ranging from 0.7 to 1.8, so $\alpha = 0.5$ lies safely within the empirical range. A sensitivity sweep over $\alpha \in [0.1, 0.9]$ on 3,000 representative slots confirms that overall SLA stays above 99% throughout, with settling times between 5 and 10 slots, indicating robustness to moderate step-size misconfiguration [10].

V. Stability and Convergence Analysis

Convergence. We define the Lyapunov function $V(t) = \sum_i (P_i^w - \bar{P}^w)^2$ as total squared pressure deviation. Since $\partial P_i^w / \partial B_i < 0$, update C1 always moves each P_i^w toward the mean $\bar{P}^w$. By a standard local contraction argument on the linearised dynamics [3], [4], each iteration satisfies:

$$V(t+1) \le (1 - \eta)\, V(t), \tag{4}$$

where $\eta = 2\alpha c(1 - \alpha \bar{P}^w)$ and $c = \min_i P_i^w / \bar{P}^w > 0$ is the minimum normalised pressure.

Settling time. This contraction holds locally near equilibrium under the linearised pressure dynamics; it is not a global guarantee across all traffic realisations. The lowest pressure ratio in the dataset is $c = 0.375$ (mMTC, minimum weight, low demand). Substituting $c = 0.375$, $\alpha = 0.5$, $\bar{P}^w = 1.0$ gives $\eta \approx 0.31$ and $t_{95} = \log(0.05)/\log(1{-}\eta) \approx 6$–$8$ slots (800 ms), consistent with the simulated settling behaviour in Fig. 3(b). This bound applies at non-idle operating points; slots with $D_i \approx 0$ are excluded since the pressure signal is negligible.

Feasibility of C2. S1 clips every candidate into $[B_i^{\min}, B_i^{\max}]$. S2a–S2b then distribute the remaining surplus after reserving all floor allocations. A feasible interior always exists and the output sums to exactly C, since $\sum_i B_i^{\min} < C < \sum_i B_i^{\max}$ (Table I: $20 < 100 < 150$).

The Lyapunov argument is best read as a local stability indicator for the linearised, unprojected dynamics under the simplified queuing model. The two-stage projection C2 introduces state-dependent clipping that resists closed-form global analysis; the guarantees are therefore intentionally conservative and are complemented by the empirical evidence in Section VII.

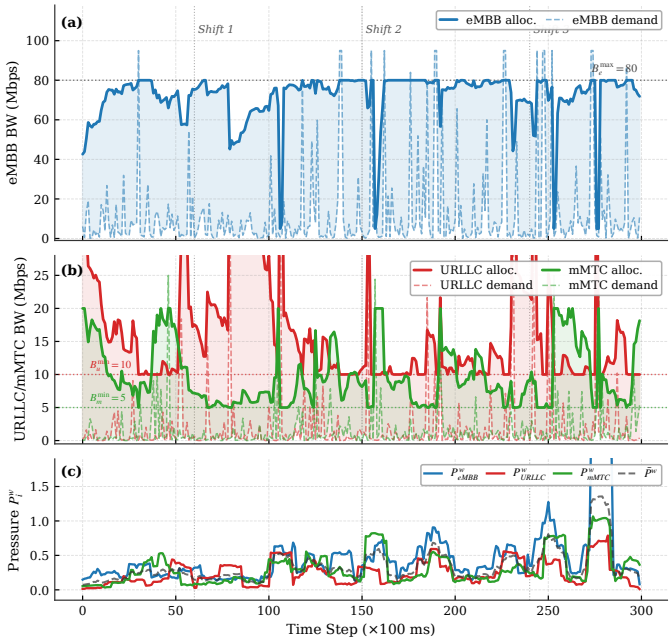

Fig. 2: 300-slot allocation trace for zone F3. *Top*: eMBB bandwidth vs. demand across three burst events. *Middle*: URLLC and mMTC bandwidths; floors hold throughout. *Bottom*: per-slice pressures converge to $\bar{P}^w$ within 6–8 slots (Section V).

VI. Dataset and Simulation Setup

No standardised 6G traffic corpus currently exists; we therefore use the Telecom Italia Big Data Challenge CDR dataset [11] as a proxy for heterogeneous mobile demand variability. The dataset records real mobile activity across Milan during November–December 2013. After cleaning, 3,670,016 records remain, partitioned into 7 spatial zones (F1–F7) of 524,288 slots each—all subdivisions of a single city-wide trace. Internet, voice, and SMS activity serve as a stress-test proxy for eMBB, URLLC, and mMTC respectively, following the volume-and-latency-sensitivity ordering used in prior studies of this corpus. CDRs expose only aggregate counts per grid cell and time bin, so the mapping is a conservative approximation rather than a precise 6G service taxonomy [1]. Mean demands are 6.38, 0.62, and 0.59 Mbps; 95th-percentile values are 37.54, 3.44, and 3.00 Mbps [11]. All zones use identical hyperparameters ($\alpha = 0.5$, $\mathbf{w}^* = [2.0, 1.5, 1.5]$) with no per-zone tuning.

VII. Results

A. Allocation Dynamics

Fig. 2 shows 300 consecutive slots from zone F3. When eMBB demand rises to 6× its mean, DPAS increases the eMBB allocation while keeping URLLC and mMTC above their respective floor constraints throughout the burst. Pressures return to near-equal equilibrium within 6–8 slots, consistent with the theoretical settling bound from Section V. Disabling C2 produces visible mMTC floor violations during the same high-load slots, confirming the necessity of the projection stage.

TABLE V: Summary results over 3,670,016 CDRs and 7 zones.

Metric	Result	Target
eMBB SLA	99.03%	≥99% ✓
URLLC SLA	99.00%	≥99% ✓
mMTC SLA	99.22%	≥99% ✓
Convergence	6–8 slots (600–800 ms)	$\leq \tau_{95}$ ✓
Complexity	$\mathcal{O}(N)$ per slot	≤1 ms (est.) ✓
Training data	None	— ✓
Stability	Lyap.-style (local)	✓

Cond. delay: eMBB 0.19 ms; URLLC 0.09 ms; mMTC 0.19 ms. Latency target based on complexity estimate only; hardware validation is future work.

TABLE VI: Per-zone SLA (%) across 7 spatial zones of the Telecom Italia Milan dataset.

Zone	eMBB	URLLC	mMTC	Mean
F1	99.06	98.83	99.12	99.00
F2	98.99	98.86	99.15	99.00
F3	99.11	98.93	99.21	99.08
F4	99.14	98.90	99.21	99.08
F5	99.02	98.73	99.08	98.94
F6	98.99	99.25	99.35	99.20
F7	98.92	99.52	99.43	99.29
All	99.03	99.00	99.22	99.09

B. Ablation Study

Reversing the gradient sign in C1 drives bandwidth away from congested slices, causing overall SLA to drop well below 99%. Replacing C2 with naïve renormalisation preserved eMBB and URLLC performance but allowed mMTC SLA to fall below the 99% target, confirming the floor-violation argument of Section IV. Reducing the mMTC weight to $w_m = 0.5$ attenuated its pressure signal during eMBB bursts and, across all seven zones, dropped mMTC SLA well below 95%; reversing the gradient sign drove overall SLA below 90%. Together, these ablations confirm that all three components are necessary for consistent SLA compliance.

C. Full-Dataset and Per-Zone Results

Tables VI and V summarise performance over all 3,670,016 records. Conditional queuing delays are 111–2660× below the corresponding SLA thresholds; the conditional delay for URLLC is 0.09 ms, and mMTC uses less than 1% of its 500 ms budget. Every zone surpasses the 99% mMTC target. The largest inter-zone spread occurs in URLLC (98.73–99.52%), reflecting spatial load heterogeneity within the Milan dataset rather than diverse deployment conditions. Fig. 3 shows $V(t)$ from cold start and conditional delays on log scale for all seven zones.

VIII. Discussion

Pressure as a state variable. Unlike a queue backlog, its P_i^w is re-computed from scratch in each slot without any history accumulated. Its deviation from the mean directly scales the bandwidth correction, yielding proportional damping at negligible overhead and keeping memory at $\mathcal{O}(N)$ words, regardless of history length.

Scope of reported SLA. All results are simulation-based estimates under a simplified M/M/1 queueing model and a

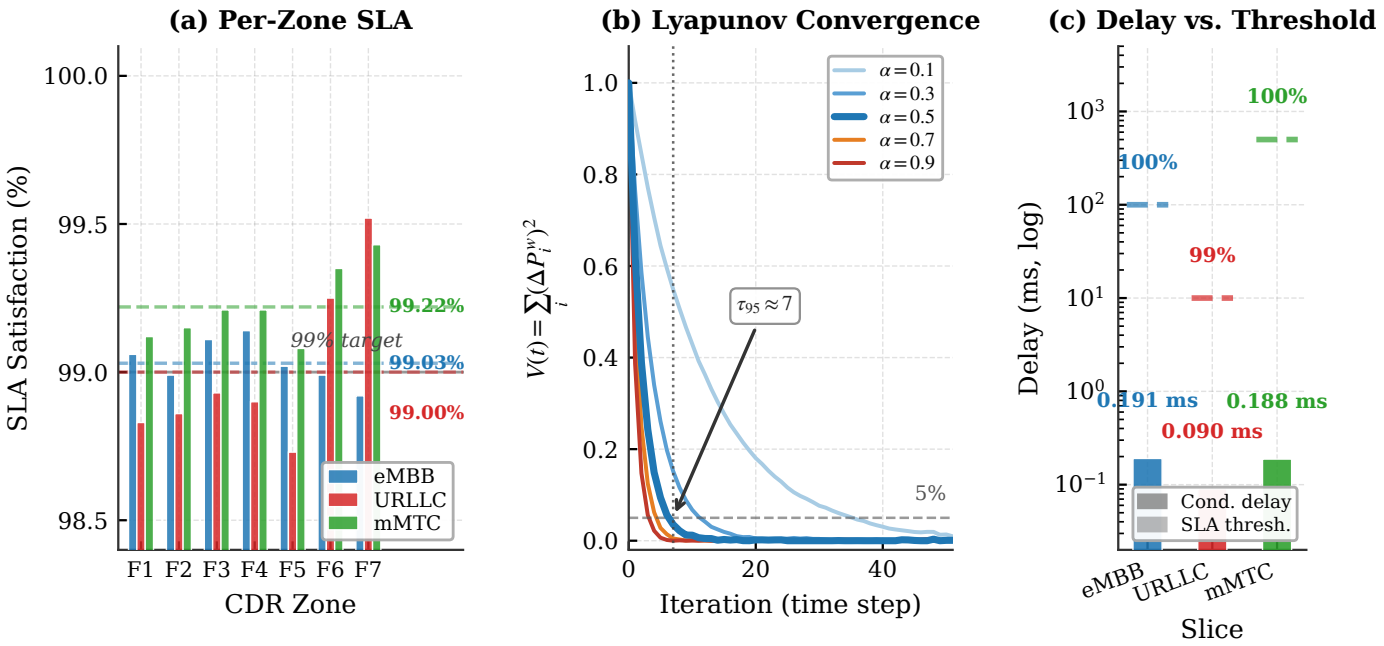

Fig. 3: Full-dataset results across 3,670,016 CDRs and 7 zones. (a) Per-zone SLA; dashed line = 99% target. (b) Lyapunov convergence for five α values; bound $\tau_{95} \leq 8$ slots. (c) Conditional delay vs. SLA threshold (log scale); delays range 111–2660× below limits.

single-cell abstraction; interference, mobility, and true URLLC burstiness are absent. They demonstrate algorithmic robustness under real mobile demand traces but do not constitute end-to-end deployment performance claims.

Limitations. The stability analysis linearises near equilibrium, and the two-stage projection C2 resists closed-form global treatment. All seven zones derive from a single city-wide dataset, so inter-zone variation reflects spatial heterogeneity within Milan rather than diverse deployment conditions. The weights $\mathbf{w}^*$ may require re-validation for dense IoT or 6G XR traffic profiles. The latency claims are grounded in algorithmic complexity alone and should be confirmed through hardware-in-the-loop trials before deployment conclusions are drawn.

IX. Conclusion and Futute Work

DPAS is a closed-form, zero-training beyond-5G slice scheduler that compresses each slice's state to a weighted demand pressure $P_i^w = w_i D_i / B_i$, enforces per-slice bandwidth floors via a two-stage projection, and, under our linearised model and simulations, typically settles within about 8 slots, all in $\mathcal{O}(N)$ operations per frame. Evaluated across 3.67 million CDRs and 7 spatial zones without per-zone tuning, DPAS achieves 99.03%, 99.00%, and 99.22% SLA for eMBB, URLLC, and mMTC respectively. The $\mathcal{O}(N)$ structure is algorithmically compatible with sub-millisecond scheduling budgets; hardware-in-the-loop trials on O-RAN testbeds will confirm this bound.

Future work includes multi-cell pressure consensus across O-RAN O-DU/O-CU, hardware-in-the-loop validation, and hierarchical NFV sub-slicing for finer granularity. A further direction is to integrate DPAS with a Layer-3 per-packet allocator such as SARA-6G [12] and neuromorphic edge architectures [13] for jointly optimised 3GPP-compliant scheduling and energy-efficient 6G IoT operation.

References

[1] 3GPP, "Service requirements for the 5G system," TS 22.261, Rel. 17, 2021.

[2] N. C. Luong et al., "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 2019.

[3] M. J. Neely, *Stochastic Network Optimization*, Morgan & Claypool, 2010.

[4] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge Univ. Press, 2004.

[5] V. Sciancalepore, K. Samdanis, X. Costa-Pérez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile traffic forecasting for maximizing 5G network slicing resource utilization," in *Proc. IEEE INFOCOM*, Atlanta, GA, USA, May 2017, pp. 1–9.

[6] I. Afolabi et al., "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2429–2453, 2018.

[7] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, Mar. 1998.

[8] 3GPP, "System architecture for the 5G system," TS 23.501, Rel. 17, 2022.

[9] L. Kleinrock, *Queueing Systems, Vol. 1*, Wiley-Interscience, 1975.

[10] J. D. Panjwani, "DPAS: Source code and CDR pipeline," GitHub, 2026. [Online]. Available: https://github.com/JAYNILiiits/DPAS-Demand-Pressure-Adaptive-Slicing

[11] Telecom Italia, "Big Data Challenge CDR dataset, Milan," *Harvard Dataverse*, 2015. [Online]. Available: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/EGZHFV

[12] J. D. Panjwani *et al.*, "SARA-6G: Standards-aligned resource allocation for 6G," in *Proc. IEEE CSNT*, 2026, pp. 146–151.

[13] J. D. Panjwani *et al.*, "Energy-efficient neuromorphic computing for 6G IoT networks: Ultra-low-power environmental sensing and edge intelligence," in *Proc. IEEE ANTS*, 2025, pp. 1–6.